Computer Vision Final Project

Yi Shao
School of Data Science
Fudan University
19307130113@fudan.edu.cn

Miaopeng Yu
School of Data Science
Fudan University
19307130203@fudan.edu.cn

Xinkai Lv
School of Data Science
Fudan University
19300180115@fudan.edu.cn

Abstract

In the first part of the project, we test an open-sourced Deeplab v3 model trained on the Cityscapes dataset on a driving video, and the resulting video is shown in the link provided below. In the second part of the project, we train three Faster R-CNN models on the VOC 2007 and 2012 datasets, and the best one uses the pre-trained Mask R-CNN's backbone. By resplitting the train/val set, this model finally reaches an mAP of 81.11% on the test set. In the third part of the project, we design the Vision Transformers with the same number of parameters as our refined ResNet-18 in the midterm assignment and train them on the CIFAR-100 dataset. The best model achieves accuracies of 69.41% and 89.88% for top-1 and top-5 metrics. We also fine-tune the pre-trained ViT on CIFAR-100. The results even outperform some of the original paper's, which achieve accuracies of 91.80% and 98.67% for top-1 and top-5 metrics, respectively.

The code's repo[1], driving video and resulting video [2], and our trained models [3] of this project are also provided at the links below.

1 Introduction

1.1 Introduction to the Three Tasks

In the first part of the project, we need to find an open-sourced semantic segmentation model trained on the Cityscapes dataset [1], and apply this model to a driving video.

In the second part of the project, we need to train the Faster R-CNN [10] model using the VOC dataset [4] using three methods. (1) Randomly initialize the weight and train the network on the VOC dataset; (2) Use the pre-trained backbone network trained on the ImageNet [5] dataset, and fine-tune the model using the VOC dataset; (3) Use the backbone network of the Mask R-CNN [7] model trained on the COCO dataset [6], and fine-tune the model using the VOC dataset.

In the third part of the project, we need to design a Transformer model with the same number of parameters as the CNNs model we used in the midterm project, and compare the results of these two models.

[1] GitHub Repo: https://github.com/Tequila-Sunrise/FDU-Computer-Vision-Final

[2] Videos: https://drive.google.com/drive/folders/1iFgFBGFRjPXXF57c9PZRKXVIl0Xgvkqb?usp=sharing

[3] Trained Models: https://github.com/Tequila-Sunrise/FDU-Computer-Vision-Final/releases/tag/publish

1.2 General Organization of this Report

Since these three tasks are not quite related, we decide to divide the report into three parts. The outline of this report is as follows. In section 2, we briefly introduce the model we use to test on the driving video, and visualize several images and the result of this video. In section 3, we describe the details of the background and implementation of the three Faster R-CNN model. In section 4, we design a Transformer model that have the same amount of parameters with the CNNs model we used in the midterm project, and compare the model of these models. In section 5, we give our conclusion for this project. We can see that we just provided a general organization here, a more detailed organization of each task is provided in 2.1, 3.1 and 4.1.

2 Test the Deeplab v3 Model on a Driving Video

2.1 Overview

In this task, we choose DeepLab v3 [3], a model trained on the Cityscapes [1] Dataset as our semantic segmentation model. Then we downloaded a driving video and then tested the model on it. And this model has a good performance on this driving video.

2.2 Cityscapes Dataset

The Cityscapes dataset [1] mainly focuses on a semantic understanding of urban street scenes, which means it includes lots of urban scenes and is labeled with each object. It is highly useful for training a model that can correctly analyze the environment, and the model can be used in many autonomous systems such as self-driving cars.

There are 5000 annotated images with fine annotations and 20000 with coarse annotations in this dataset. The pictures in this dataset are taken in 50 cities throughout the daytime for several months, and with good or medium weather conditions. It labeled 30 classes of objects, which were divided into 8 groups, so as to gain a brief knowledge of the environment.

2.3 A Brief Introduction to Deeplab v3

DeepLab v3 [3] is a semantic segmentation architecture that improves upon DeepLab v2 [2] with several modifications. When designing the modules, atrous convolution was employed in cascade or in parallel to capture multi-scale context by adopting multiple atrous rates, which can handle the problem of segmenting objects at multiple scales.

The procedure of plain atrous convolution is as shown in Figure 1. Compared with normal convolution, this method can be more effective for semantic image segmentation, and allows us to repurpose ImageNet pre-trained networks to extract denser feature maps by removing the downsampling operations from the last few layers and upsampling the corresponding filter kernels.

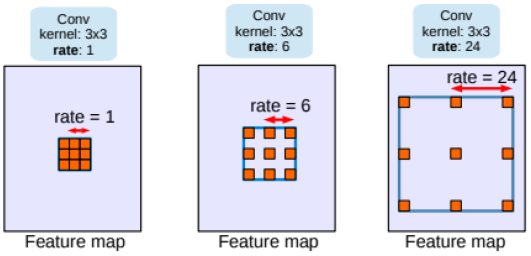


Figure 1: Atrous convolution

Traditional DCNN is constructed with standard convolution and pooling operations, making the output feature map smaller after each block operation, which is useful in capturing long-range information. And by adding atrous convolution layers, we can preserve spatial resolution and make a deeper network by capturing features at each scale. This procedure is shown in Figure 2. We can see that the field of view of the filter becomes wider, which is required for better semantic segmentation results.

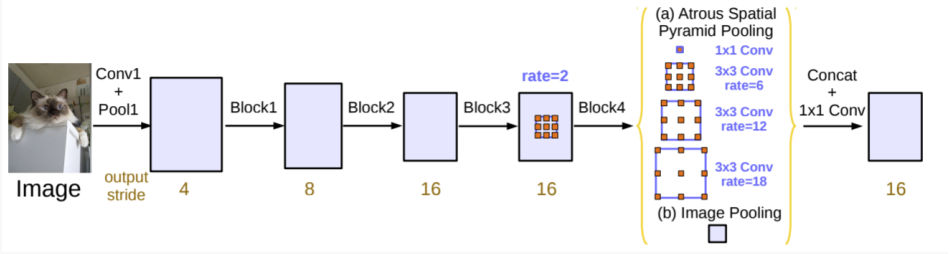
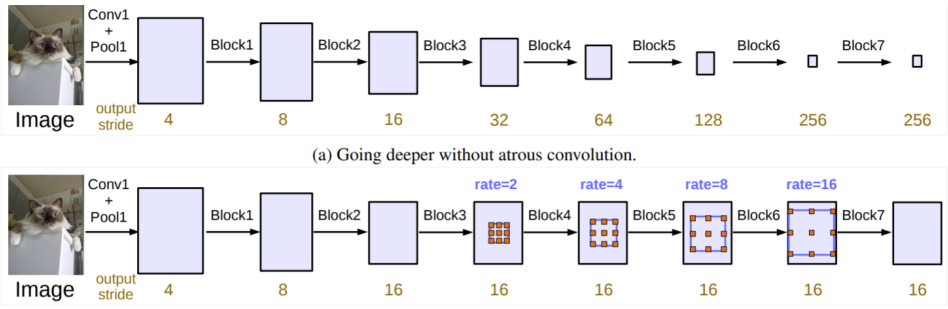


Figure 2: DCNN with and without Atrous Convolution

Also, in order to capture objects at multiple scales, we can do atrous convolution in different sizes and then combine them together. This technique is called Atrous Spatial Pyramid Pooling (ASPP) method. In this way, the model can gain a larger field of view and is able to distinguish features at many scales.

Generally, the architecture of DeepLab v3 is shown in Figure 3. It first uses a backbone network (VGG, DenseNet, ResNet) to extract the feature, then applies atrous convolution in the last few blocks to control the size of the feature map. On top of these, an ASPP network is added to classify each pixel corresponding to its classes. The output is then passed through a 1x1 convolution to get the actual size of the image, and will form the final segmented mask of the image.



(a) Going deeper without atrous convolution.

(b) Going deeper with atrous convolution. Atrous convolution with $rate > 1$ is applied after block3 when $output\_stride = 16$.

Figure 3: DeepLab V3 Model Architecture

The model was later trained on Cityscapes dataset. The 5000 images with high quality pixel-level annotations were then divided into 2975, 500, and 1525 for the training, validation, and test sets, respectively. After setting output stride to 8, employing multi-scale inputs, and adding left-right flipped inputs, the model has achieved an mIOU of 79.30%.

2.4 Visualization of the Segmentation Results

Part of the results of our segmentation is shown below in Figure 4. We can see that the model is quite good at distinguishing cars, roads and humans. Full video is uploaded to google drive, and the link can be found at the footnote 2 of page 1.

3 Train Faster R-CNN on the VOC 2007 and 2012 Dataset

3.1 Overview

In the second task, we train the Faster R-CNN on the VOC 2007 and 2012 datasets using three different methods. (1) Randomly initialize the weight and train the network on the VOC dataset; (2) Use the pre-trained backbone network trained on the ImageNet [5] dataset, and fine-tune the model using the VOC dataset; (3) Use the backbone network of the Mask R-CNN [7] model trained on the COCO dataset [6], and fine-tune the model using the VOC dataset. And we also show the loss and mAP curve in the training set and the test set.

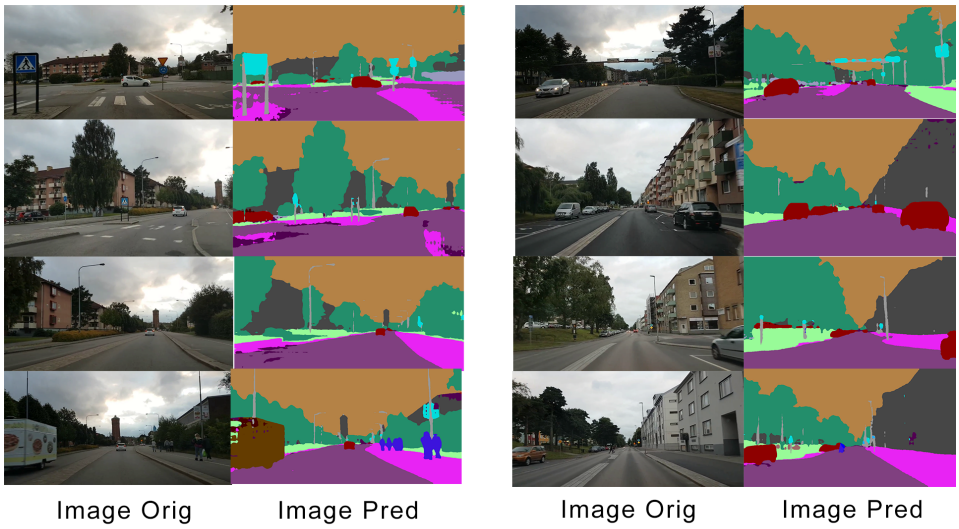


Figure 4: Segmentation Results

Since after combining the VOC 2007 and 2012 datasets, the validation set is quite large. Therefore, we resplit the train/val set, and get a larger mAP than using the original dataset settings. The details of this process will be shown later. Finally, we get an mAP of 58.17%, 80.39%, and 81.11% for these three methods in the test set, respectively.

The organization of section 3 is as follows. In section 3.2, we introduce the VOC dataset, and how we divide the dataset into different sets. In section 3.3, we introduce the model Faster R-CNN. In section 3.4, we show the implementation details of training the three models like batch size, learning rate, optimizer, etc. In section 3.5, we report our results of these three models, and visualize the training and testing loss and mAP curve in the test set.

3.2 VOC Dataset and its Division

The PASCAL VOC dataset is a dataset for image recognition. And it is fundamentally a supervised learning problem that a training set of labeled images is provided. The twenty object classes include person, bird, cat, cow, dog, horse, sheep, aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, TV/monitor. Each image in this dataset has pixel-level segmentation, bounding box annotations, and object class annotations. This dataset has been widely used as a benchmark for object detection tasks. [4]

For VOC 2007, it contains 9,963 images, 5,011 of them are divided into the train/val set, and 4,952 of them are divided into the test set. For VOC 2012, it only makes the train/val set containing 17,125 images public. [4]

We use both the VOC 2007 and 2012 for training the Faster R-CNN model, but instead of using the original split - train/val set are divided into training and validation set 1:1, we resplit the train/val set of the VOC 2012 dataset, and keep the split of the VOC 2007 dataset unchanged due to the large volume of the validation samples. We resplit the train/val set of the VOC 2012 dataset, make 90% of them the training sample, and 10% of them as validation sample, and add up the training and validation sample on the VOC 2007 dataset. After that, we can make the training set : validation set approximately to 0.8:0.2, which is shown in Table 1.

Table 1: Resplit of the VOC Dataset

data	# train	# val	# test
VOC 2007	2501 (49.9%)	2510 (50.1%)	4952
VOC 2012	15412 (90.0%)	1713 (10.0%)	0
Total	17913 (80.9%)	1713 (19.1%)	4952

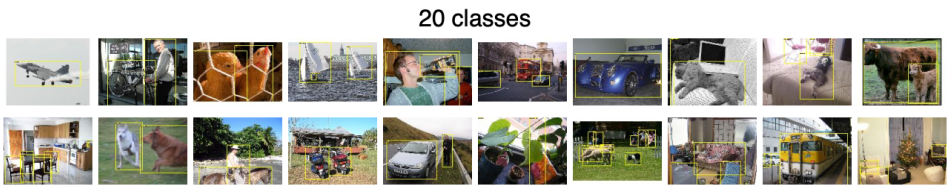
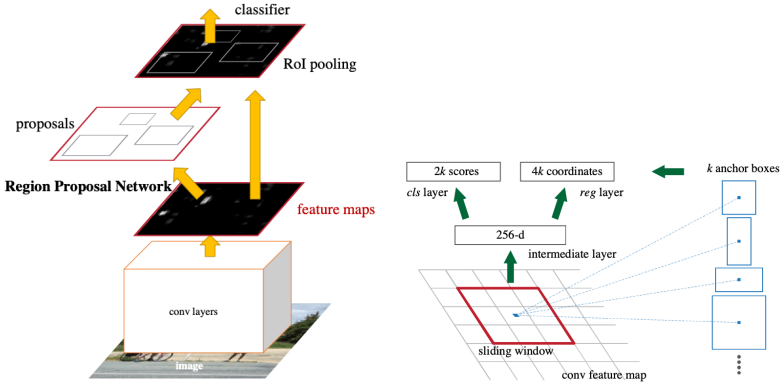


Figure 5: VOC Examples [4]

3.3 A Brief Introduction to Faster R-CNN

Although region-based CNNs are computationally expensive as the originally developed in [8], their cost has been drastically reduced thanks to sharing convolutions across proposals like Fast R-CNN, which achieves near real-time rates using very deep networks, if we ignore the time spent on region proposals [9]. However, the time spent on region proposals is too long. So in Faster R-CNN, the author introduces a Region Proposal Network (RPN) that shares full-image convolution features with the detection network, and it can generate proposals with various scales and aspect ratios to tell the object detection (Fast R-CNN) where to look. And the RPN module serves as the "attention" of Faster R-CNN.

The architecture of Faster R-CNN is shown in Figure 6(a). It consists of 2 modules: RPN for generating the region proposals, and Fast R-CNN for detecting objects in the proposed regions.



(a) Architecture of Faster R-CNN (b) Architecture of RPN

Figure 6: Architecture of Faster R-CNN and RPN

After extracting features from the image using the backbone network. The next step of Faster R-CNN is generating region proposals using RPN. The RPN works on the output feature map returned from the last convolutional layer shared with the Fast R-CNN. Based on a rectangular window of size $n \times n$, a sliding window passes through the feature map. For each window, several anchor boxes are generated, and will be later filtered based on their "objectness score". Then we adjust these anchor boxes to get the proposal boxes.

After that, for all regions proposed in the image, a fixed-length feature vector is extracted from each region using the ROI Pooling layer. Then the extracted feature vectors are then classified using the Fast R-CNN and used to adjust the proposal box. Finally, the class scores of the detected objects in addition to their bounding boxes are returned. [10]

And the loss function shown in the original paper is shown below.

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*),$$

where p_i is the predicted probability of anchor i being an object. t_i is a vector representing the 4 parameterized coordinates of the predicted bounding box. And the elements with a $*$ is the ground-

truth, and t and t^* are normalized to [-1, 1], L_{cls} is the log loss over two classes, $L_{reg}(t_i, t_i^*) = L_1(t_i, t_i^*)$. And N_{cls} is mini-batch size, N_{reg} is the anchor locations, and λ is typically set to 10 to balance the weight between the two part of the loss function. [10]

3.4 Implementation Details

While training, we re-scale the images such that their shorter side is 600 pixels, since multi-scale feature extraction may improve accuracy but does not exhibit a good speed-accuracy trade-off [15]. The backbone we use in Faster R-CNN is pre-trained ResNet-50. And we choose to use pre-trained networks is for faster and better convergence of the network. For anchors, we use 3 scales with box areas of $128^2, 256^2, 512^2$ pixels, and 3 aspect ratios of 1:1, 1:2 and 2:1.

To reduce the redundancy, we adopt non-maximum suppression (NMS) on the propoosal regions based on their cls scores, we fix the IoU threshold for NMS at 0.3, and keep 12,000 anchor boxes according to their scores while training, and 6,000 anchor boxes while testing. After that, we keep 600 proposal boxes according to their scores while training, and 300 proposal boxes while testing. And we set the confidence to be 0.5.

The optimizer we choose for this model is SGD with a weight decay of 1×10^{-5} and a momentum of 0.9. We start with a learning rate of 0.01 and reducing the learning rate using cosine annealing. And we train the model for 150 epochs. The epoch settings for the three tasks are a bit different.

3.4.1 Model A: Training from Scratch

We initialize all parameters in the model using normal initialization. There is no freeze training process, all of the 150 epochs are used to train the whole Faster R-CNN together, and the batch size is 4.

3.4.2 Model B: Using pre-trained Backbone Model on ImageNet

Since the pre-trained models in module "torchvision.model" are pre-trained using the ImageNet dataset, we can use the models in this module directly. And in the first 50 epochs, we will freeze the backbone, and train the remained network only, the batch size we choose in these 50 epochs is 8. After 50 epochs, we unfreeze the backbone and train the whole Faster R-CNN together, the batch size we choose in the last 100 epochs is 4.

3.4.3 Model C: Using Backbone of Mask R-CNN pre-trained on COCO Dataset

We can download the model from mmlab[4], and load the backbone of this pre-trained model. And the setting of the freezing and unfreezing process is the same as model B.

We train the three models using one NVIDIA RTX 3090.

3.5 Results and Visualization of the Training Process

We train and evaluate our model on the resplit PASCAL VOC 2007 and 2012 dataset. And the final mAP of the three models mentioned above on the test set is shown in Table 2.

Table 2: Result of Different Models on the VOC 2007+2012 Dataset. Backbone is ResNet-50.

index	backbone	mAP0.5
a	Randomly Inistialize	58.17
b	pre-trained on ImageNet	80.39
c	Backbone of Mask R-CNN pre-trained on COCO dataset	81.11

And the more detailed results are shown in Table 3.

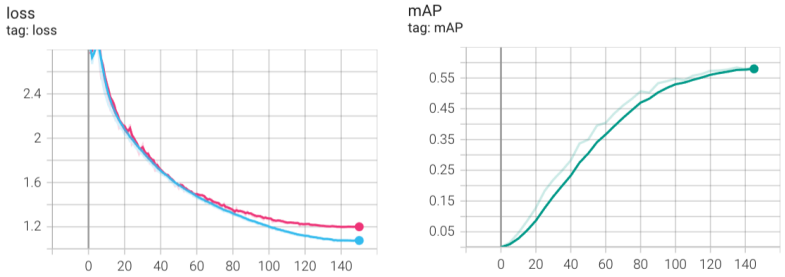
The following plots Figure 7 show the loss curve for the training set and the test set, and the mAP curve for the test set for the three models. And for model b and c, the dramatica rise in loss is due to the unfreezing process, where the loss in the backbone is also taken into account.

[4] `https://github.com/open-mmlab/mmdetection/tree/master/configs/mask_rcnn`

Table 3: Detailed Result of Different Models on the VOC Dataset

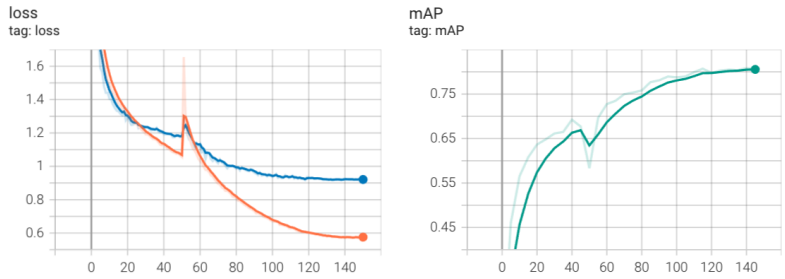
model index	data	mAP0.5	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow
a	07+12	58.17	60.68	68.55	49.74	46.94	24.52	69.47	74.08	65.63	35.14	56.23
b	07+12	80.39	81.13	86.87	81.99	74.84	62.22	89.50	88.07	91.20	62.61	84.50
c	07+12	81.11	83.35	86.32	81.64	74.65	68.58	88.85	91.22	90.85	67.62	82.46

table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
56.23	62.02	77.98	69.98	76.93	28.39	56.73	57.33	68.63	58.27
75.89	90.15	90.20	86.04	83.89	54.26	84.85	74.87	87.69	76.97
76.05	85.93	90.08	85.71	86.76	53.53	84.95	80.61	85.42	77.56



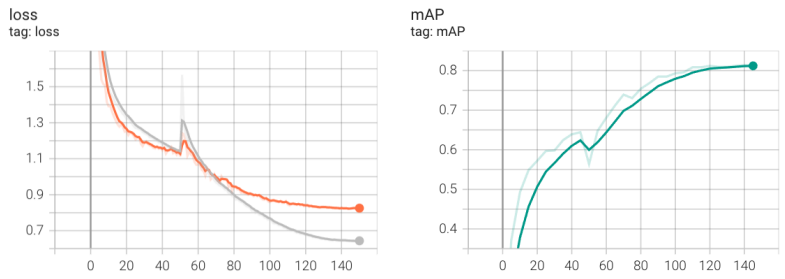
(a) Loss of model A

(b) mAP of model A



(c) Loss of model B

(d) mAP of model B



(e) Loss of model C

(f) mAP of model C

Figure 7: Curves of Training Faster R-CNN using VOC dataset with TensorBoard

By putting the loss curve together, we can get the plot Figure 8.

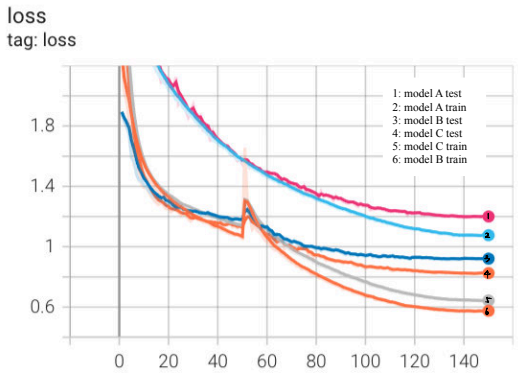


Figure 8: Loss of the Three Models

As we can see in the plot, model A has the largest training loss and the largest test loss. It is because the weight is randomly initialized at the beginning, so it will take a longer time to train. And due to the decaying step size, it is more likely to fall to a local minimum during the training process.

For model C, although the training loss is slightly larger than the one in model B, the test loss is smaller than the one in model B. It is a quite interesting outcome since we expected that the backbone in model C is pre-trained to fit the object detection task better than in model B, the training loss of model C should also be smaller than the one of model B. However, it is probably because the backbone of model C has seen a lot of data (COCO dataset), it is less likely to overfit the data in the VOC dataset, so the training loss is slightly worse while the test loss is better.

4 Train Vision Transformer on the CIFAR-100 Dataset

4.1 Overview

In the last task, we design a Vision Transformer (ViT) model with almost the same number of parameters as our refined ResNet-18 in the midterm assignment and train it on the CIFAR-100 dataset [11] from scratch. The open-source frameworks we mainly referred to are including the vanilla Vision Transformer [15] and Vision Transformer for small-size datasets [16]. In the training process, random augmentation is performed to improve the model effect, which is different from the fixed data augmentation methods we apply in the previous assignments like CutOut [12], Mixup [13], or CutMix [14]. In the end, we will compare their class accuracy on the test set.

4.2 Dataset CIFAR-100 and its Division

The CIFAR-100 dataset (Canadian Institute for Advanced Research, 100 classes) is a labeled subset of the 80 million tiny images dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. And it has 100 classes containing 600 images each, 500 of them are training images, and 100 of them are testing images. So for the 60,000 images, 50,000 of them are assigned to the training set, and 10,000 of them are assigned to the test set. Also, each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs). And we are using the "fine" label.

4.3 A Brief Introduction to Vision Transformer

An overview of the model architecture is depicted in Figure 10. The Vision Transformer, or ViT, is a model for image classification that employs a Transformer-like architecture over patches of the image. An image is split into fixed-size patches, each of them is then linearly embedded, position embeddings are added, and the resulting sequence of vectors is fed to a standard Transformer encoder. In order to perform classification, the standard approach of adding an extra learnable “classification token” to the sequence is used.

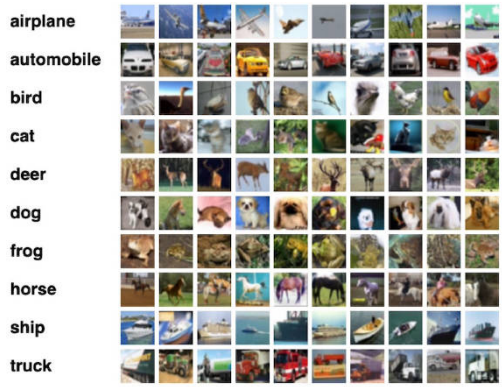


Figure 9: CIFAR-100 Examples [11]

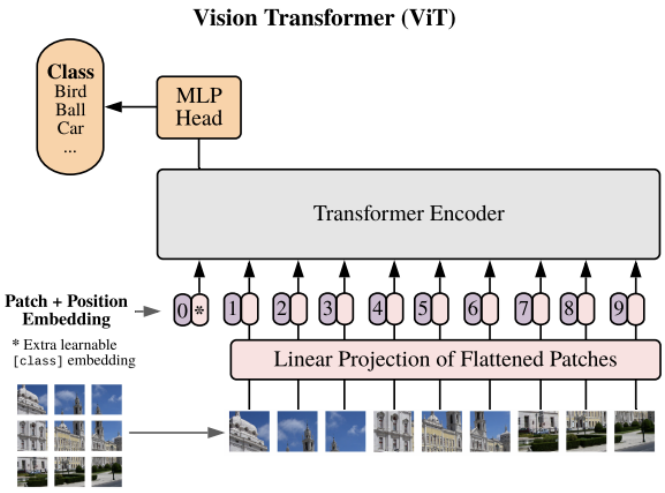


Figure 10: Architecture of Vision Transformer

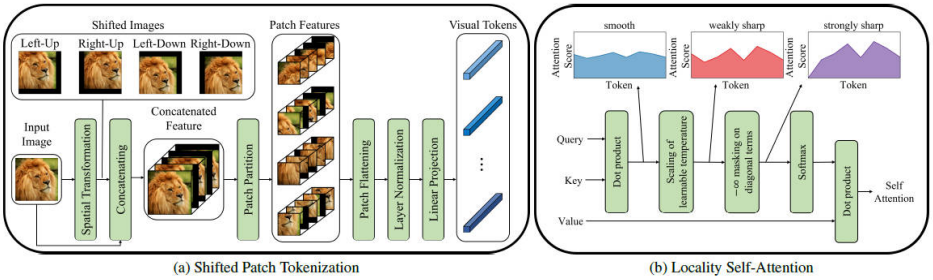
The standard Transformer receives as input a 1D sequence of token embeddings. To handle 2D images, we have to reshape the image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ into a sequence of flattened 2D patches $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where (H, W) is the resolution of the original image, C is the number of channels, (P, P) is the resolution of each image patch, and $N = HW/P^2$ is the resulting number of patches, which also serves as the effective input sequence length for the Transformer. The Transformer uses constant latent vector size D through all of its layers, so we flatten the patches and map to D dimensions with a trainable linear projection.

Position embeddings are added to the patch embeddings to retain positional information. Here we just use the standard learnable 1D position embeddings, since the paper points out that no significant gains in performance are observed from using more advanced 2D-aware position embeddings. The resulting sequence of embedding vectors serves as input to the encoder.

The Transformer encoder consists of alternating layers of multiheaded self-attention and MLP blocks. Layernorm (LN) is applied before every block, and residual connections after every block. The MLP contains two layers with a GELU non-linearity.

$$
\begin{aligned}
\mathbf{z}_0 &= \left[\mathbf{x}_{\text{class}}\,; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}\right] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{\left(P^2 \cdot C\right) \times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D} \\
\mathbf{z}'_\ell &= \text{MSA}\left(\text{LN}\left(\mathbf{z}_{\ell-1}\right)\right) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L \\
\mathbf{z}_\ell &= \text{MLP}\left(\text{LN}\left(\mathbf{z}'_\ell\right)\right) + \mathbf{z}'_\ell, \qquad \ell = 1 \ldots L \\
\mathbf{y} &= \text{LN}\left(\mathbf{z}_L^0\right)
\end{aligned}
$$

However, the high performance of the ViT results from pre-training using a large-size dataset such as JFT-300M, and its dependence on a large dataset is interpreted as due to low locality inductive bias. Thus another work proposes Shifted Patch Tokenization (SPT) and Locality Self-Attention (LSA) (illustrated in Figure 11), which effectively solve the lack of locality inductive bias and enable it to learn from scratch even on small-size datasets. Moreover, SPT and LSA are generic and effective add-on modules that are easily applicable to various ViTs.



(a) Shifted Patch Tokenization (b) Locality Self-Attention

Figure 11: Architecture of Vision Transformer

4.4 Implementation Details

First, we adjust the hyper-parameters of ViT in order the maintain the equivalent model volume as the previous refined ResNet-18 model. After some fine-tuning, we come up with the ViT model of dimension set to 512, depth set to 6, attention head is set to 8, and MLP dimension set to 1024. Their number of parameters are both nearly 11M.

In the training process, we follow what has been done in the midterm assignment, the input image is 32x32 randomly cropped from a zero-padded 40x40 image or its horizontally flipping, and normalization is performed. What's more, then we abandon the data augmentation methods employed before and use random augmentation methods [17] instead, this is one of the automated augmentation strategies which can eliminate not only the obstacle in the separate search phase, which increases the training complexity and may substantially increase the computational cost, but also the difficulty to adjust the regularization strength based on the model or dataset size.

Referring to the ViT paper, we initialize the learning rate with 8e-4 and adopt a cosine annealing scheduler. The models are trained for up to 200 epochs. Furthermore, the criterion is chosen as cross-entropy with label smoothing, while the optimizer is Adam with a weight decay of 1e-4, and dropout is applied as well.

Models are trained on one NVIDIA RTX 3090.

4.5 Results and Visualization

We train and evaluate our model on the CIFAR-100 dataset. Among all the checkpoints, the best test accuracies of all models are listed in Table 4 for comparison.

From the results shown in Table 4, we can observe that if training from scratch on the small-size dataset, ViT with SPT and LSA structures does gain improvement on classification effects compared with the vanilla ViT, while they both perform worse than the refined ResNet-18. This outcome is consistent with the ViT paper, which indicates that Transformers lack some of the inductive biases inherent to CNNs, such as translation equivariance and locality, therefore do not generalize well when trained on insufficient amounts of data.

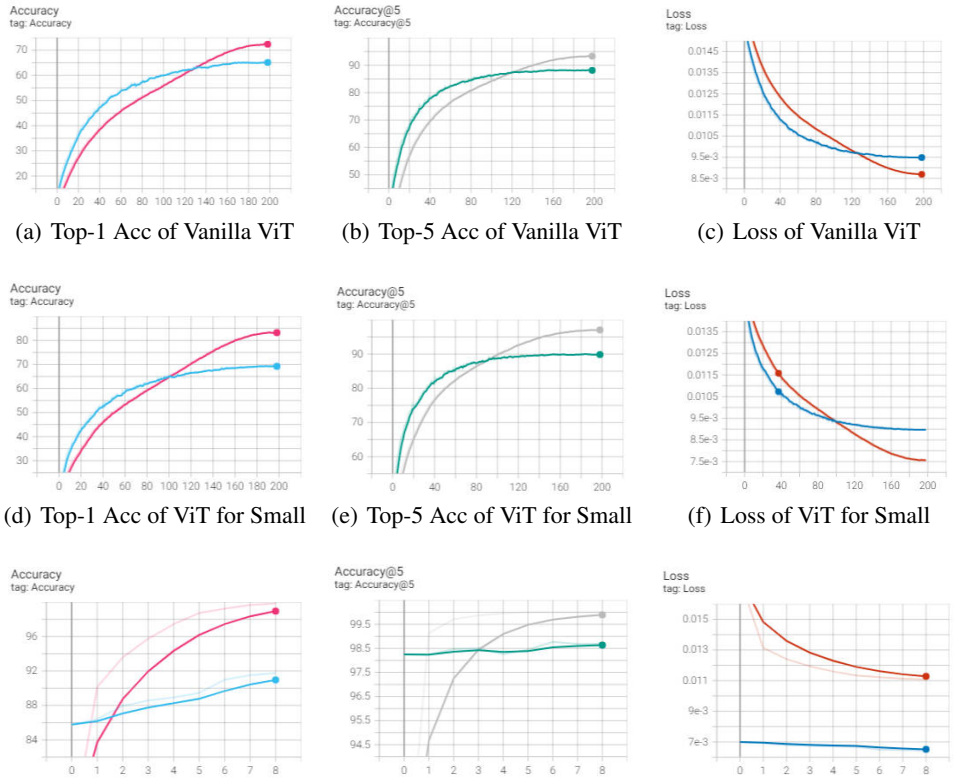
Table 4: Result of Different Models on the CIFAR-100 Dataset

model	#params	top-1 acc	top-5 acc
Refined ResNet-18	11.8M	79.53%	94.22%
Vanilla ViT	12.7M	65.32%	88.16%
ViT for Small-Size Datasets	12.8M	69.41%	89.88%
ViT (pre-trained on ImageNet-21K)	87M	91.80%	98.67%

Then we fine-tune the pre-trained ViT-base-patch16-224 model on CIFAR-100, and to our expectations, this model presents unprecedentedly excellent results, which even outperforms the paper's output.

The phenomenon above indicates Vision Transformer's superiority in scalability, large scale training finally trumps inductive bias, and it also attains close results compared to convolutional networks of the same number of parameters while requiring substantially fewer computational resources to train.

All the tensorboard plots can be found below in Figure12.



(a) Top-1 Acc of Vanilla ViT (b) Top-5 Acc of Vanilla ViT (c) Loss of Vanilla ViT

(d) Top-1 Acc of ViT for Small (e) Top-5 Acc of ViT for Small (f) Loss of ViT for Small

(g) Top-1 Acc of Pre-trained ViT (h) Top-5 Acc of Pre-trained ViT (i) Loss of Pre-trained ViT

Figure 12: Curves of Training Vision Transformer using CIFAR-100 dataset with TensorBoard

5 Conclusion

In this project, we test a driving video on a trained DeepLab v3 model, train Faster R-CNN in three different ways on the VOC datasets, and train ViT on the CIFAR-100 dataset. With the help of pre-trained models and some interesting tricks, some of our models even have a better performance than the ones provided by the original authors.

References

[1] Cordts, M., Omran, M., Ramos, S., Scharwächter, T., Enzweiler, M., Benenson, R., ... & Schiele, B. (2015, June). The cityscapes dataset. In CVPR Workshop on the Future of Datasets in Vision (Vol. 2). sn.

[2] Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, 40(4), 834-848.

[3] Chen, L. C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587.

[4] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2), 303-338.

[5] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.

[6] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.

[7] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

[8] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

[9] Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[10] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.

[11] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.

[12] DeVries, T., & Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552.

[13] Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412.

[14] Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 6023-6032).

[15] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

[16] Lee, S. H., Lee, S., & Song, B. C. (2021). Vision Transformer for Small-Size Datasets. arXiv preprint arXiv:2112.13492.

[17] Cubuk, E. D., Zoph, B., Shlens, J., & Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 702-703).